



**Sistema Informático para el Control de Activo Fijo y Préstamo
de Libros**

Manual del Programador

Versión: 1.0

INDICE

1	INTRODUCCION	3
1.1	Objeto.....	3
1.2	Alcances.....	3
2	ESTRUCTURA DEL PROYECTO	4
2.1	Descripción de las carpetas.....	5
2.2	Repositorios	6
2.2.1	Connect.....	6
2.2.2	Restore.....	7
2.2.3	Activo	8
2.2.4	Administradores	14
2.2.5	Autores	16
2.2.6	Categoría.....	16
2.2.7	Detalle.....	16
2.2.8	Editoriales.....	17
2.2.9	Encargado	17
2.2.10	Institución	17
2.2.11	Mantenimiento.....	18
2.2.12	Préstamo Activos	18
2.2.13	Préstamo Libros.....	19
2.2.14	Proveedor.....	19
2.2.15	Recuperación de contraseña	20
2.2.16	Usuarios.....	20
2.2.17	Libros.....	22
3	GLOSARIO	23



1 INTRODUCCION

1.1 *Objeto*

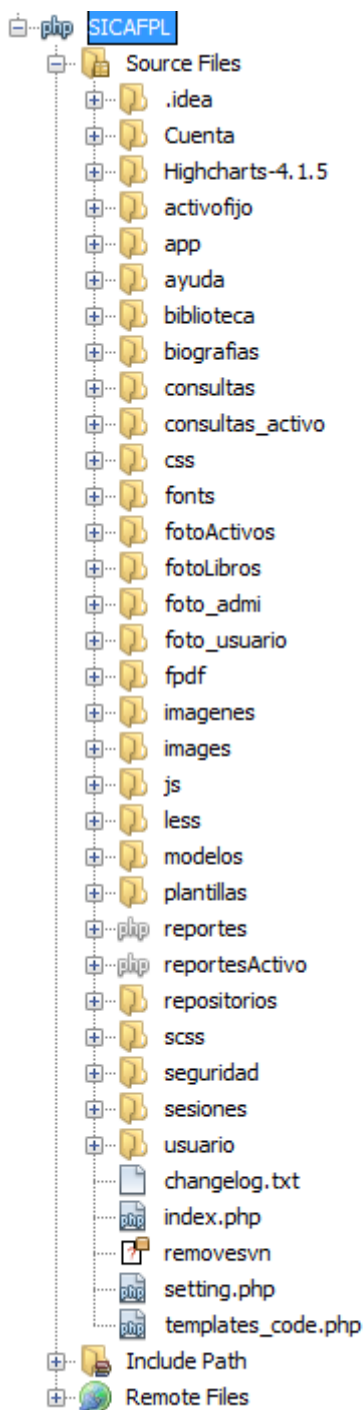
Explicar la estructura del proyecto informático para que el lector pueda hacer modificaciones que ayuden a un mejor funcionamiento del sistema.

Dar a conocer los métodos utilizados en el sistema informático, para que sirven y donde se usan.

1.2 *Alcances*

Este documento va dirigido a toda persona que tenga los conocimientos básicos del lenguaje de programación PHP en su versión 7.2, y que desee hacer modificaciones en el funcionamiento del sistema informático.

2 ESTRUCTURA DEL PROYECTO



2.1 Descripción de las carpetas

Raíz: Ubicación principal del proyecto, aquí se encuentra el índice.

.idea, Highcharts-4.1, fpdf: Estas carpetas contienen librerías externas que ayudan al funcionamiento del sistema.

Cuenta: Contiene los archivos que controlan la cuenta del administrador.

activo_fijo: contiene todos los archivos del módulo de activo fijo.

App: contiene los archivos de configuración del sistema como: conexión a la base de datos, y el inicio de sesión del sistema.

Ayuda: contiene los archivos que muestran la ayuda del sistema.

Biblioteca: contiene los archivos del módulo de biblioteca.

Biografías: contiene las biografías que se suben durante el registro de los autores.

Consultas: contiene las consultas del módulo de biblioteca.

Consultas_activo: contiene las consultas del módulo de activo fijo

Css: contiene todas las hojas de estilos css.

Fonts: contiene fuentes e iconos.

fotosActivos: contiene las fotos que se cargan al registrar activos.

fotoLibros: contiene las fotos que se cargan al registrar libros.

Foto_admini: contiene las fotos que se cargan al registrar administradores.

Foto_usuario: contiene las fotos que se cargan al registrar usuarios.

Imágenes e images: contiene otras imágenes usadas en el sistema.

Js: contiene los scripts de javascript.

Less: contiene archivos .less utilizados por bootstrap.

Modelos: contiene todos los objetos utilizados en el sistema.

Plantillas: contiene secciones similares de cada vista, las cuales son llamadas en cada archivo con la sentencia `include_once`.

Reportes: contiene los reportes del módulo de biblioteca.

reportesActivo: contiene los reportes del módulo de activo fijo.

Repositorios: contiene todos los métodos que interactúan con la base de datos, separados por modelos.

Scss: archivos .scss utilizados por bootstrap.

Seguridad: contiene los archivos del módulo de seguridad.

Sesiones: contiene el cierre de sesión.

Usuario: contiene los archivos del módulo de usuario.

2.2 Repositorios

2.2.1 Connect

```
<?php
//establecemos las variables globales que seran ocupadas por el backup y restore
error_reporting(E_PARSE);
//Nombre de usuario de mysql
const USER = "root";
//Servidor de mysql
const SERVER = "localhost";
//Nombre de la base de datos
const BD = "diseno1";
//Contraseña de mysql
const PASS = "";
//Carpeta donde se almacenaran las copias de seguridad
//include_once '../backup/';
const BACKUP_PATH = 'C:/Dropbox/backup/';

//configuramos la zona horaria para que sea la de El Salvador
date_default_timezone_set('America/El_Salvador');
```

```
class SGBD
{
    //Funcion para hacer consultas a la base de datos
    public static function sql($query)
    {
        $con = mysqli_connect(SERVER, USER, PASS, BD);
        ///usamos esto para que sea nomenclatura latina
        mysqli_set_charset($con, "utf8");
        if (mysqli_connect_errno()) {
            printf("Conexion fallida: %s\n", mysqli_connect_error());
            exit();
        } else {
            //empezamos haciendo el commit para guardar la base de datos
            mysqli_autocommit($con, false);
            mysqli_begin_transaction($con, MYSQLI_TRANS_START_WITH_CONSISTENT_SNAPSHOT);
            if ($consul = mysqli_query($con, $query)) {
                if (!mysqli_commit($con)) {
                    print("Falló la consignación de la transacción\n");
                    exit();
                }
            } else {
                mysqli_rollback($con);
                echo "Falló la transacción";
                exit();
            }
            return $consul;
        }
    }
}
```

```
//Funcion para limpiar variables que contengan inyeccion SQL
public static function limpiarCadena($valor)
{
    $valor = addslashes($valor);
    $valor = str_ireplace("<script>", "", $valor);
    $valor = str_ireplace("</script>", "", $valor);
    $valor = str_ireplace("SELECT * FROM", "", $valor);
    $valor = str_ireplace("DELETE FROM", "", $valor);
    $valor = str_ireplace("UPDATE", "", $valor);
    $valor = str_ireplace("INSERT INTO", "", $valor);
    $valor = str_ireplace("DROP TABLE", "", $valor);
    $valor = str_ireplace("TRUNCATE TABLE", "", $valor);
    $valor = str_ireplace("--", "", $valor);
    $valor = str_ireplace("^", "", $valor);
    $valor = str_ireplace("[", "", $valor);
    $valor = str_ireplace("]", "", $valor);
    $valor = str_ireplace("\\\\", "", $valor);
    $valor = str_ireplace("=", "", $valor);
    return $valor;
}
?>
```

2.2.2 Restore

```
<?php
$titulo1 = 'Restaurar';
include_once '../plantillas/cabecera.php';
include_once '../plantillas/menu.php';
include '../repositorio_Connet.php';

//estos metodos se ejecutaran cuando el administrador halla confirmado su contraseña, luego de darle click al boton
//restaurar que se encuentra en el archivo backup_restore que se encuentra en la carpeta seguridad

//esto es para evitar inyecciones sql
$restorePoint = SGBD::limpiarCadena($_REQUEST['restorePoint']);

//recuperamos el punto de restauracion
$sql = explode(";", file_get_contents($restorePoint));

//establecemos banderas
$totalErrors = 0;
//limite de espera de la sentencia sql ejecutada
set_time_limit(60);

//establecemos que trabajaremos con mySql
//y enviamos los valores de las variables globales que recuperamos de el repositorio_Connet.php
$con = mysqli_connect(SERVER, USER, PASS, BD);

//establecemos la codificacion latina
mysqli_set_charset($con, "utf8");

//ejecutamos la sentencia
$con->query("SET FOREIGN_KEY_CHECKS=0");
```

```
//recorremos los valores de el punto de recuperacion del sistema
for ($i = 0; $i < (count($sql) - 1); $i++) {
    if ($con->query($sql[$i] . ";") {

    } else {
        $totalErrors++;
    }
}

$con->query("SET FOREIGN_KEY_CHECKS=1");
$con->close();
if ($totalErrors <= 0) {
    //si todo se realiza con exito mandamos los mensajes de confirmacion
    echo '<script>swal({
        title: "Exito",
        text: "Se restaura el sistema a un estado anterior, vuelva a iniciar su sesión para continuar!",
        type: "success",
        confirmButtonText: "ok",
        closeOnConfirm: false
    },function () {location.href="../sesiones/cerrar.php";});</script>';
} else {
    echo '<script>swal({
        title: "Exito",
        text: "Se restaura el sistema a un estado anterior, vuelva a iniciar su sesión para continuar!",
        type: "success",
        confirmButtonText: "ok",
        closeOnConfirm: false
    },function () {location.href="../sesiones/cerrar.php";});</script>';
}

include_once '../plantillas/pie_de_pagina.php';
```

2.2.3 Activo

```
public static function insertar_activo($conexion, $activo) { //funcion para insertar activos a la base de datos
    //recibe la conexion y un objeto tipo activo que contiene los datos a registrar
    $activo_insertado = false;
    if (isset($conexion)) { //comprueba que la conexion esta abierta
        try {
            //asignamos los datos a nuevas variables para tener mas orden
            $codigo_activo = $activo->getCodigo_activo();
            $codigo_tipo = $activo->getCodigo_tipo();
            $codigo_proveedor = $activo->getCodigo_proveedor();
            $codigo_detalle = $activo->getCodigo_detalle();
            $codigo_administrador = $activo->getCodigo_administrador();
            $estado = $activo->getEstado();
            $observacion = "";
            $foto = $activo->getFoto();
            $fecha = $activo->getFecha_adquisicion();
            $precio = $activo->getPrecio();
            // finaliza la asignacion

```

```
//sentencia que se ejecuta para ingresar activo a la base
$sql = 'INSERT INTO activos(codigo_activo,codigo_tipo, codigo_proveedor, codigo_detalle, codigo_administrador, fecha_adquisicion, precio, estado, foto, observacion)
        ' values (:codigo_activo,:codigo_tipo, :codigo_proveedor , :codigo_detalle, :codigo_administrador, :fecha, :precio, :estado, :foto , :observacion)';

//estos son alias para que PDO pueda trabajar
$sentencia = $conexion->prepare($sql);
$sentencia->bindParam(':codigo_activo', $codigo_activo, PDO::PARAM_STR);
$sentencia->bindParam(':codigo_tipo', $codigo_tipo, PDO::PARAM_STR);
$sentencia->bindParam(':codigo_proveedor', $codigo_proveedor, PDO::PARAM_STR);
$sentencia->bindParam(':codigo_detalle', $codigo_detalle, PDO::PARAM_INT);
$sentencia->bindParam(':codigo_administrador', $codigo_administrador, PDO::PARAM_INT);
$sentencia->bindParam(':fecha', $fecha, PDO::PARAM_STR);
$sentencia->bindParam(':precio', $precio, PDO::PARAM_STR);
$sentencia->bindParam(':estado', $estado, PDO::PARAM_INT);
$sentencia->bindParam(':observacion', $observacion, PDO::PARAM_STR);
$sentencia->bindParam(':foto', $foto, PDO::PARAM_STR);

$activo_insertado = $sentencia->execute(); // si tuvo exito la ejecucion del sql $activo_insertado sera true
} catch (PDOException $ex) { // en caso de error en ejecutar el sql muestra el siguiente mensaje
    echo '<script>swal("No se puede realizar el registro activo", "Favor revisar los datos e intentar nuevamente" . $ex->getMessage() . "", "warning");</script>';
    print 'ERROR: ' . $ex->getMessage();
}
}
```



```
public static function Lista_activo($conexion) { //funcion para recuperar y mostrar todos lo activos de la base de datos
// en la ventana inventario
$resultado = ""; //aqui se guardan todos los datos
if (isset($conexion)) {
    try {
        $sql = "SELECT * from activos ORDER BY activos.codigo_activo ASC";
        $resultado = $conexion->query($sql); //se asigna todos los datos a la variable
    } catch (PDOException $ex) {
        print 'ERROR: ' . $ex->getMessage();
    }
}
return $resultado; //se envia una lista con todos los activos que es recorrida mediante un foreach
}
```

```
public static function Lista_activo_codBarra($conexion) { //para listar activos en el reporte codigos de barra - act
$resultado = ""; //aqui se guardan todos los datos
if (isset($conexion)) {
    try {
        $sql = "SELECT
activos.codigo_activo ,
activos.codigo_tipo as cod,
categoria.nombre as tipo
FROM
activos
INNER JOIN categoria ON activos.codigo_tipo = categoria.codigo_tipo
GROUP BY
activos.codigo_tipo";
        $resultado = $conexion->query($sql); //se asigna todos los datos a la variable
    } catch (PDOException $ex) {
        print 'ERROR: ' . $ex->getMessage();
    }
}
return $resultado; // envia la lista con todos los activos que es recorrida mediante un foreach
}
```

```
public static function Lista_activo_tipo($conexion, $tipo) { //retorna la lista de activos de un tipo en especifico
//recibe la conexion y el codigo tipo de activo para filtrar
$resultado = ""; //aqui se guardan todos los datos
if (isset($conexion)) {
    try {
        $sql = "SELECT
activos.codigo_activo as cod
FROM
activos
WHERE
activos.codigo_tipo ='$tipo' AND
activos.estado = 1";
        $resultado = $conexion->query($sql); //se asigna todos los datos a la variable
    } catch (PDOException $ex) {
        print 'ERROR: ' . $ex->getMessage();
    }
}
return $resultado; // envia la lista con todos los activos que es recorrida mediante un foreach
}
```

```

    public static function lista_activo_inventario($conexion, $cual) {
        //lista activos para las consultas inventario, activos dañados y extraviados
        //si la variable $cual esta vacia ejecuta normal el sql
        $resultado = "";
        if($cual=="3"){//si $cual = 3 filtra solamente los activos dañados
            $cual='WHERE activos.estado = 3';
        }
        if($cual=="4"){//si $cual = 4 filtra solamente los activos extraviados
            $cual='WHERE activos.estado = 4';
        }
        if (isset($conexion)) {
            try {
                $sql = "SELECT
                categoria.nombre as tipo,
                activos.codigo_activo as cod,
                CONCAT(proveedores.nombre,' ') as proveedor,
                CONCAT(administradores.nombre,' ',administradores.apellido) as admin,
                activos.estado as e,
                activos.precio as p,
                activos.fecha_adquisicion as f,
                activos.observacion as o
                FROM
                activos
                INNER JOIN categoria ON activos.codigo_tipo = categoria.codigo_tipo
                INNER JOIN administradores ON activos.codigo_administrador = administradores.codigo_administrador
                INNER JOIN proveedores ON activos.codigo_proveedor = proveedores.codigo_proveedor
                $cual
                ";
                $resultado = $conexion->query($sql);
            } catch (PDOException $ex) {
                print 'ERROR: ' . $ex->getMessage();
            }
        }
    }

```

```

    public static function lista_activo_mas($conexion) { //retorna lista de activos mas prestados
        $resultado = "";

        if (isset($conexion)) {
            try {
                $sql = "SELECT
                prestamo_activos.codigo_pactivo,
                activos.codigo_activo as cod,
                categoria.nombre as tipo,
                (select count(*) from movimiento_activos where movimiento_activos.codigo_activo =cod) as veces
                FROM
                prestamo_activos
                INNER JOIN movimiento_activos ON movimiento_activos.codigo_pactivo = prestamo_activos.codigo_pactivo
                INNER JOIN activos ON movimiento_activos.codigo_activo = activos.codigo_activo
                INNER JOIN categoria ON activos.codigo_tipo = categoria.codigo_tipo
                GROUP BY
                cod
                ORDER BY
                veces desc
                ";
                $resultado = $conexion->query($sql);
            } catch (PDOException $ex) {
                print 'ERROR: ' . $ex->getMessage();
            }
        }
        return $resultado; //enviamos la lista
    }

```

```

public static function lista_activo_baja($conexion) {///retorna los activos dados de baja
    $resultado = "";
    if (isset($conexion)) {
        try {
            $sql = "SELECT
categoria.nombre AS tipo,
actvos.codigo_activo AS codigo,
actvos.precio AS p,
actvos.estado AS e,
actvos.observacion AS o,
(CONCAT(administradores.nombre, ' ',administradores.apellido)) as nombre,
actvos.fecha_adquisicion as f

FROM
actvos
INNER JOIN categoria ON actvos.codigo_tipo = categoria.codigo_tipo
INNER JOIN administradores ON actvos.codigo_administrador = administradores.codigo_administrador
WHERE
actvos.estado = 0
";// el estado 0 es de los actios dado de baja
            $resultado = $conexion->query($sql);
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
    return $resultado;//enviamos la lista
}

```

```

public static function lista_activo2($conexion) {///retorna la lista de activos disponibles para prestar
    $resultado = "";
    if (isset($conexion)) {
        try {
            $sql = "SELECT * from actvos where estado='1' ORDER BY actvos.codigo_activo ASC";
            // estado 1 es disponible
            $resultado = $conexion->query($sql);
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
    return $resultado;//enviamos la lista
}

```

```

public static function lista_activo_mantenimiento($conexion) {///lista los activos disponibles y los dañados.
//se utiliza en el buscador para agragar activos a la tabla de mantenimiento
    $resultado = "";
    if (isset($conexion)) {
        try {
            $sql = "SELECT * from actvos where actvos.estado = 1 or actvos.estado = 3 ORDER BY actvos.codigo_activo ASC";
            $resultado = $conexion->query($sql);
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
    return $resultado;//enviamos la lista
}

```

```
public static function lista_activo_mantenimiento2($conexion) { //retorna los activos dañados a la seccion de mantenimineto
    $resultado = "";
    if (isset($conexion)) {
        try {
            $sql = "SELECT * from activos where activos.estado = 3 ORDER BY activos.codigo_activo ASC";
            $resultado = $conexion->query($sql);
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
    return $resultado; //enviamos la lista
}
```

```
public static function lista_activo3($conexion, $cant) { //lista los activos disponibles para prestar
    //utilizada en el buscador para agregar activos a la tabla de prestamo
    $resultado = "";
    if (isset($conexion)) {
        try {
            $sql = "SELECT
                    activos.codigo_activo
                    FROM
                    activos
                    WHERE
                    activos.estado = 1

                    LIMIT '$cant'";
            $resultado = $conexion->query($sql);
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
    return $resultado; //enviamos la lista
}
```

```
public static function actualizar_activo($conexion, $activo, $codigo_original) { //funcion para modificar los dats de un activo
    $activo_insertado = false;

    if (isset($conexion)) {
        try {
            //los datos que se modifican son el encargado del activo y la foto
            $codigo_administrador = $activo->getCodigo_administrador();
            $foto = $activo->getFoto();

            if ($foto != "") { //si el campo foto no esta vacio se actualizan codigo del adminstrador a cargo del activo y la foto
                $sql = "UPDATE activos set codigo_administrador='$codigo_administrador', foto = '$foto' where codigo_activo='$codigo_original'";
            } else { //el campo foto esta vacio solo se actualiza el codigo del administrador
                $sql = "UPDATE activos set codigo_administrador='$codigo_administrador' where codigo_activo='$codigo_original'";
            }

            $sentencia = $conexion->prepare($sql);
            $activo_insertado = $sentencia->execute();

            $accion = "Se actualizaron los datos del activo " . $codigo_original;
            //variable saccion se envia a la bitacora del sistema
            self::insertar_bitacora($conexion, $accion); //insertamos en la bitacora

        } catch (PDOException $ex) {
            echo "<script>swal('Excelente!', 'hubo pido '$sql' ', 'success');</script>";
            print 'ERROR: ' . $ex->getMessage();
        }
    }
}
```

```

public static function Lista_activo3($conexion, $cant) { //lista los activos disponibles para prestar
//utilizada en el buscador para agregar activos a la tabla de prestamo
$resultado = "";
if (isset($conexion)) {
    try {
        $sql = "SELECT
                activos.codigo_activo
                FROM
                activos
                WHERE
                activos.estado = 1

                LIMIT '$cant'";
        $resultado = $conexion->query($sql);
    } catch (PDOException $ex) {
        print 'ERROR: ' . $ex->getMessage();
    }
}
return $resultado; //enviamos la lista
}

```

```

public static function actualizar_activo($conexion, $activo, $codigo_original) { //funcion para modificar los datos de un activo
    $activo_insertado = false;

    if (isset($conexion)) {
        try {
            //los datos que se modifican son el encargado del activo y la foto
            $codigo_administrador = $activo->getCodigo_administrador();
            $foto = $activo->getFoto();

            if($foto != "") { //si el campo foto no esta vacio se actualizan codigo del administrador a cargo del activo y la foto
                $sql = "UPDATE activos set codigo_administrador='$codigo_administrador', foto = '$foto' where codigo_activo='$codigo_original'";
            } else { //el campo foto esta vacio solo se actualiza el codigo del administrador
                $sql = "UPDATE activos set codigo_administrador='$codigo_administrador' where codigo_activo='$codigo_original'";
            }

            $sentencia = $conexion->prepare($sql);
            $activo_insertado = $sentencia->execute();

            $accion = "Se actualizaron los datos del activo " . $codigo_original;
            //variable $accion se envia a la bitacora del sistema
            self::insertar_bitacora($conexion, $accion); //insertamos en la bitacora

        } catch (PDOException $ex) {
            echo "<script>swal('Excelente!', 'hubo pido '$sql' ', 'success');</script>";
            print 'ERROR: ' . $ex->getMessage();
        }
    }
}

```

```

public static function obtener_activo($conexion, $codigo) { //obtenemos todos los datos de un activo segun el codigo recibido
    $resultado = "";
    if (isset($conexion)) {
        try {
            $sql = "SELECT * from activos where codigo_activo='$codigo'";
            $resultado = $conexion->query($sql);
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
    return $resultado;
}

```

```
public static function obtener_estadoActivo($conexion, $cod) { //obtenermos el estado del activo segun el codigo

    if (isset($conexion)) {
        try {

            $sql = "SELECT
                activos.estado
            FROM
                activos
            WHERE
                activos.codigo_activo = '$cod'";
            foreach ($conexion->query($sql) as $row) {
                $r = $row[0];
            }
            return $r;
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
}
```

```
public static function obtener_nactivo($conexion, $cod) { //se obtiene el numero de activos que estan registrado segun el tipo

    if (isset($conexion)) {
        try {

            $sql = "SELECT
                COUNT( activos.codigo_tipo)
            FROM
                activos
            WHERE
                activos.codigo_tipo = '$cod'"; //filtra para tener solo de un tipoS
            foreach ($conexion->query($sql) as $row) {
                $r = $row[0];
            }
            return $r;
        } catch (PDOException $ex) {
            print 'ERROR: ' . $ex->getMessage();
        }
    }
}
```

2.2.4 Administradores

///esta funcion sera utilizada desde la la vista registro_administrador que se encuentra alojada en la carpeta seguridad, recibe como parametros: conexion, que sera ocupada para acceder a la base de datos, como segundo parametro utilizara un objeto de tipo administrador en el cual estan los datos que fueron enviados desde la vista

```
public static function insertar_administrador($conexion, $administrador) {...123 lines }
```

///este metodo lo utilizamos en el inicio de sesion, devuelve un objeto tipo administrador con el cual le verificamos si el usuario y contraseña coinciden con los ingresados

```
public static function obtener_administrador($conexion, $codigo_administrador) {...23 lines }
```

///este metodo es utilizado en la vista listar administrador para asi poder visualizar todos los administradores con el fin de que sean eliminados o modificados, dejamos fuera de esta lista al administrador que actualmente esta con sesion activa y al administrador principal (admin01)

```
public static function lista_administradores($conexion, $codigo) {...37 lines }
```

///este metodo es utilizado en la vista administradores_eliminados que se encuentra en la carpeta seguridad se utiliza para listar todos los administradores que han sido eliminados (administradores con estado 0) recibe como parametro la conexion a la base de datos

```
public static function lista_administradores_eliminados($conexion) {...36 lines }
```

```

//este metodo es utilizado en la vista eliminar_administrador que se encuentra en la carpeta seguridad
//es utilizada cuando al eliminar un administrador, deben de transferir los activos que estan a su cargo
//tiene como restriccion al administrador actual y a los administradores que previamente hallan sido eliminados
//recibe como parametro la conexion a la base de datos y al administrador que se desea eliminar
public static function lista_administradores_para_baja($conexion, $codigo) {...39 lines }

//este metodo es utilizado por la vista editar_mis_datos que se encuentra en la carpeta Cuenta
//recibe como parametros la conexion a la base de datos, el administrador con los datos que se van a a
//y como ultimo a la nueva clase
public static function actualizarClave($conexion, $codigo_administrador, $clave) {...17 lines }

//este metodo es utilizado en la vista editar administrador que se encuentra en la carpeta seguridad
//resive como parametro la conexion a la base de datos, un objeto de tipo administrador que contiene los datos
//que se quieren actualizar, recibe tambien el codito del administrador que se desea modificar, y como ultimo
//parametro recibe la pass del actual administrador como medida de seguridad
public static function actualizar_administrador($conexion, $administrador, $codigo_original, $verificacion) {...98 lines }

//este metodo es utilizado desde la vista eliminar_administrador, recibe como parametro la conexion a la bae
// de datos, un objeto de tipo administrador del cual recuperaremos el motivo por el que se eliminara al administrador,
//el codigo del administrador a eliminar , y una pass que se comparara en la del administrador con sesion activa
public static function eliminar_administrador($conexion, $administrador, $codigo_eliminar, $verificacion) {...59 lines }

//este utilizado en el index, en el caso que el administrador inicie su sesion con el correo
//recibe como parametros: la conexion a la base de datos, el email con el que se quiere ingresar

public static function obtener_email($conexion, $email) {...19 lines }

//esta funcion es utilizada por otras funciones, tiene la mision de obtener los datos de un administrador en
//especifico, recibe como parametro: la conexion a la base de datos y el codigo del administrador
//retorna como resultado un objeto del tipo administrador con los datos del administrador si se encuentran
public static function obtener_administrador_actual($conexion, $codigo) {...36 lines }
//este metodo es ocupado desde la vista editar_mis_datos desde la carpeta Cuenta
//recibe como parametro la conexion a la base de datos, un objeto de tipo de administrador con todos los
//datos del administrador que se desea actualizar, un una pass que se ocupara para verificar por cuestiones de seguridad
public static function actualizar_mis_datos($conexion, $administrador, $verificacion) {...102 lines }
//esta funcion es utiliza al final de cada registro,modificacion y eliminacion se ocupa para guardar la informacion
//de los cambios en la bitacora
//recibe como parametros la conexion a la base de datos, y un string el cual describe la accion realizada
public static function insertar_bitacora($conexion, $accion) {...21 lines }
//esta funcion es utilizada para obtener el numero de administradores registrados en la base de datos
//como unico parametro recibe la conexion a la base de datos
public static function numero_administradores($conexion) {...18 lines }

//esta funcion es eutilizada desde el index, para verificar si la pass introducida es correcta
public static function verificar_pass($conexion, $verificacion) {...37 lines }
//esta funcion es utilizada desde la vista eliminar_administrador que se encuentra en la carpeta seguridad
//es utilizada para trasladar los activos de un administrador que se eliminara, hacia uno que este activo
public static function actualizar_activos_administradir($conexion, $codigo_administrador1, $codigo_administrador2) {...17 lines }

//esta funcion es utilizada por la funcion registro de administrador, verifica si el dui ingresado
//ya se encuentra registrado en la base de datos, recibe como parametros la conexion a la base de datos
// y el dui introducido
public static function verifica_dui($conexion, $dui) {...16 lines }

```

2.2.5 Autores

```
<?php
class Repositorio_autores {
    //Este metodo se encarga de insertar los autores en la base de datos
    //es llamado en el archivo registro_b.php ubicado en la carpeta biblioteca
    public static function insertarAutor($conexion, $autor) {...31 lines }
    //Este metodo obtiene el ultimo autor registrado en la base de datos
    public function ObtenerUltimo($conexion) {...15 lines }
    //este metodo retorna una lista de los autores
    //usado en modificar.php y catalogo de autores
    public static function ListaAutores($conexion) {...12 lines }
    //Metodo que modifica los datos de un autor
    public static function editarAutor($conexion, $autor) {...32 lines }
    //inserta en la bitacora la fecha y la accion que se hizo aqui
    static function insertar_bitacora($conexion, $accion) {...22 lines }
}

?>
```

2.2.6 Categoria

```
<?php

class Repositorio_categoria {
    //para insertar una nueva categoria o tipo de activo
    public static function insertar_categoria($conexion, $categoria) {...27 lines }
    //lista todos los tipos de activos registrados
    public static function Lista_categorias($conexion) {...27 lines }
    //obtenemos el nombre del tipo de activo
    public static function obtener_categoria($conexion, $cod) {...15 lines }
    //retorna un nuevo codigo de tipo de activo
    public static function obtener_newcod_categoria($conexion) {...27 lines }
    //obtenemos el nombre del tipo de activo
    public static function obtener_nombre_categoria($conexion, $cod) {...16 lines }
```

2.2.7 Detalle

```
<?php

class Repositorio_detalle {
    //insertamos detalles de activos a la base de datos
    public static function insertar_detalle($conexion, $detalle) {...41 lines }
    //obtenemos el codigo del ultimo detalle insertado
    public static function obtener_ultimo_detalle($conexion) {...15 lines }
    //obtenemos los detalles de un activo
    public static function obtener_detalle($conexion, $codigo_detalle) {...25 lines }
    //actualiza los detalles de un activo
    public static function actualizar_detalle($conexion, $detalle, $codigo_original) {...45 lines }
```


2.2.8 Editoriales

```
<?php

/**
 *
 */
class Repositorio_editorial {
    //metodo que inserta las editoriales
    public static function insertarEditorial($conexion, $editorial) {...30 lines }
    //metodo que obtiene la ultima editorial insertada
    public function ObtenerUltimo($conexion) {...15 lines }
    //metodo que retorna la lista de las editoriales
    public function ListaEditorial($conexion) {...12 lines }
    //metodo que edita los datos de una editorial
    public static function editarEditorial($conexion, $editorial) {...32 lines }
```

2.2.9 Encargado

```
<?php

class Repositorio_encargado {
    //inserta nuevo encargado a la base de datos
    public static function insertar_encargado($conexion, $encargado) {...37 lines }
    //obtiene datos de encargado segun el $codigo_encargado
    public static function obtener_encargado($conexion, $codigo_encargado) {...20 lines }
    //lista con los datos de los encargados
    public static function Lista_encargado($conexion) {...19 lines }
```

2.2.10 Institución

```
<?php
class Repositorio_institucion{

    ///esta funcion es utilizada en la vista registro_usuario, para llenar la lista de las
    //institucionales a la que puede pertenecer un usuario
    //recibe como parametro la conexion a la base de datos
    //y retorna la lista de instituciones
    public static function Lista_institucion($conexion) {...29 lines }

    //esta funcion es utilizada en la vista alumnos_institucion se utiliza para cargar los valores que se agregaran
    //a los graficos de dicha vista
    //recibe como parametro la conexion a la base de datos y retorna una lista con el nombre y el codigo de la
    //institucion
    public static function usuarios($conexion) {...28 lines }

    //esta funcion es utilizada en la vista alumnos_institucion se utiliza para cargar todos los datos a la
    //grafica de alumnos por institucion
    //recibe como parametro la conexion a la base de datos y el codigo de la institucion
    // retorna la cantidad de usuarios por institucion
    public static function usuario_por_institucion($conexion , $institucion) {...35 lines }

}

?>
```

2.2.11 Mantenimiento

```
<?php

/**
 *
 */
class Repositorio_mantenimiento {
    //retorna una lista con los mantenimientos registrados
    public static function ListaMantAct($conexion) {...19 lines }
    //inserta mantenimiento a la bae de datos
    public static function GuardarMantAct($conexion, $mant) {...22 lines }
    //guarda los codigos de activos que fueron a mantenimiento en la tabla de movimiento
    public static function GuardarActivos($conexion, $codAct, $codMant) {...19 lines }
    //guarda los codigos de encargados que realizaron el mantenimiento en la tabla de movimiento
    public static function GuardarEncargados($conexion, $codAct, $codMant) {...19 lines }
    //lista los codigos de encargados que realizaron el mantenimiento segun $codMant
    public static function ListarEncargados($conexion, $codMant) {...18 lines }
    //obtenems el ultimo codigo de mantenimiento registrado
    public static function obtenerUltimoMant($conexion) {...16 lines }
    //obtenemos los codigos de activos que fueron a mantenimiento segun el $codigoMant
    public static function obtenerActivos($conexion, $codigoMant) {...22 lines }
```

2.2.12 Préstamo Activos

```
<?php

/**
 *
 */
class Repositorio_prestamoact {

    public static function ListaPrestamosAct($conexion) {...29 lines }
    //devuelve una lista de los prestamos de activos registrados ordenada segun al fecha de devolucion vencida
    public static function ListaActPrestamos($conexion) {...33 lines }
    //guarda prestamode activo en la base de datos
    public static function GuardarPrestamoAct($conexion, $prestamo) {...21 lines }
    //guarda activos de un prestamo en la tabla de movimiento
    public static function GuardarActivos($conexion, $prestamo, $libro) {...26 lines }
    //recuperamos el ultipo codigo de prestamo registrado
    public static function obtenerUltimoPact($conexion) {...16 lines }
    //actualiza los datos del prestamo y queda finalizado
    public static function Finalizar($conexion, $codigo, $motivo) {...15 lines }
    //actualiza datos de prestamo fecha y obseraciones
    public static function Actualizar($conexion, $fecha, $observaciones, $cod) {...13 lines }
    //actualiza el estdo de un activo geu estaba en prestamo
    public static function ActualizarActivo($conexion, $cod, $estado, $observacion) {...27 lines }
    //obtenemos los datos de un prestamo de activo
    public static function obtenerPact($conexion, $codigoPact) {...36 lines }
    //obtenemos la lista de codigos de activos de un prestamo
    public static function obtenerListActP($conexion, $codigoP) {...25 lines }

}
```

2.2.13 Préstamo Libros

```
<?php

/**
 *
 */
class Repositorio_prestamolib {
    //retorna una lista de los prestamos pendientes
    public static function ListaPrestamos($conexion) {...31 lines }
    //retorna los libros incluidos en un prestamo
    public static function ListaLibrosPrestamo($conexion, $codigo) {...28 lines }
    //registra los prestamos en la base de datos
    public static function GuardarPrestamo($conexion, $prestamo) {...35 lines }
    //registra los libros incluidos en el prestamo, en una tabla de movimiento
    public static function GuardarLibros($conexion, $prestamo, $libro) {...34 lines }
    //obtiene el ultimo prestamo registrado
    public static function obtenerUltimo($conexion) {...16 lines }
    //finaliza un prestamo
    public static function Finalizar($conexion, $codigo, $motivo) {...27 lines }
    //actualiza la fecha de devolucion de un prestamo
    public static function Actualizar($conexion, $codigo, $fecha) {...27 lines }
    //cambia el estado de los libros incluidos en un prestamo
    public static function cambiarEstado($conexion, $codigo_libro, $estado) {...27 lines }
}

?>
```

2.2.14 Proveedor

```
<?php

class Repositorio_proveedor {
    //inserta proveedor en base de datos
    public static function insertar_proveedor($conexion, $proveedor) {...35 lines }

    //lista los proveedores registrados
    public static function Lista_proveedores($conexion) {...27 lines }

    public static function insertar_bitacora($conexion, $accion) {...21 lines }
}

?>
```

2.2.15 Recuperación de contraseña

```
<?php
class RepositorioRecuperacion {
    //registra una peticion de cambio de contrasena
    public static function registrarPeticon($conexion, $codigoAdmin, $urlSecreta) {...17 lines }
    //obtiene una peticion segun la url secreta
    public static function obtenerPeticon($conexion, $url_secreta) {...19 lines }
    //obtiene peticion segun codigo de administrador
    public static function obtenerPeticonEmail($conexion, $codigoAdmin) {...19 lines }
    //elimina una peticion despues de haber realizado el cambio de contrasena
    public static function eliminarPeticon($conexion, $codigoAdmin) {...14 lines }
}
```

2.2.16 Usuarios

```
<?php
class Repositorio_usuario {
    //esta funcion es ocupada desde la vista registro_usuario que esta en la carpeta usuario
    //es utilizada para el registro de usuarios
    //recibe como parametro la conexion a la base de datos, y un objeto de tipo usuario con los datos que se
    //deseen registrar
    public static function insertar_usuario($conexion, $usuario) {...105 lines }
    //este metodo es utilizado por la funcion registro de usuarios,
    //devuelve el numero de usuarios registrados en la base de datos
    public static function numero_de_usuarios($conexion) {...32 lines }

    ///esta funcion es utilizada desde la vista listar_usuario que se encuentra en la carpeta de de usuarios
    //devuelve un array de tipo usuario con todos los usuarios que esten activos (estado = 1)
    public static function lista_usuarios($conexion) {...41 lines }

    //esta funcion es ocupada por el archivo expediente_usuario que se encuentra en la carpeta consultas, que a la
    //vez se encuentra en la carpeta usuario, es utilizada para listar a todos los usuarios para generar reportes
    public static function lista_usuarios_completa($conexion) {...39 lines }

    //esta funcion es utilizada por la vista usuarios eliminados que se encuentra en la carpeta usuario
    //lista a todos los usuarios que han sido dados de baja (estado = 0)
    //devuelve un array de tipo usuario
    //recibe como parametros solamente la conexion a la base de datos
    public static function lista_usuarios_eliminados($conexion) {...39 lines }
```

Manual de Actualización Programador

```
//esta funcion es utilizada por la vista editar_usuario que se encuentra en la carpeta usuario
//se utiliza para actualizar datos de un usuario en especifico
//recibe como parametros: la conexion a la base de datos, un objeto de tipo usuario con los datos para actualizar
//y el carnet del usuario que se quiere actualizar
public static function actualizar_usuario($conexion, $usuario, $carnet) {...76 lines }

//esta funcion es ocupada por la vista eliminar_usuario que se encuentra en la carpeta usuario
//actualiza el estado del usuario a inactivo (estado = 0)
//recibe como parametro la conexion a la base de datos, un objeto de tipo usuario que contiene la
//informacion sobre la eliminacion y el carnet del usuario que se desea eliminar
public static function eliminar_usuario($conexion, $usuario, $carnet) {...59 lines }

public static function insertar_bitacora($conexion, $accion) {...19 lines }

//este metodo se ocupara en la vista eliminar_usuario que se encuentra en la carpeta usuario
//sirve para verificar si el usuario a eliminar cuenta o no con prestamos de activos sin finalizar
//retorna un booleano segun sea el caso
//recibe como parametros la conexion de la base de datos y el carnet del usuario que se desea eliminar
public static function comprobar_prestamos_activos($conexion, $usuario) {...24 lines }

//este metodo se ocupara en la vista eliminar_usuario que se encuentra en la carpeta usuario
//sirve para verificar si el usuario a eliminar cuenta o no con prestamos de libros sin finalizar
//retorna un booleano segun sea el caso
//recibe como parametros la conexion a la base de datos y el carnet del usuario que se desea eliminar
public static function comprobar_prestamos_libros($conexion, $usuario) {...25 lines }

//esta funcion es utilizada por la vista lista_carnet_alumno que se encuentra en la carpeta consulta, que
//a su vez se encuentra en la carpeta usuario, se utiliza para seleccionar todos los datos de un usuario en especifico
//recibe como parametros la conexion a la base de datos y el carnet del usuario deseado
public static function usuario_seleccionado($conexion, $codigo) {...54 lines }

//esta funcion es para saber cual fue el ultimo usuario ingresado en la base de datos
//recibe como parametro la conexion a la base de datos
public static function ultimo_usuario_insertado($conexion) {...22 lines }

//esta funcion es ocupada en la vista expediente_usuario que esta en la carpeta consulta, que se encuentra en la
//carpeta usuario
//es utilizada para listar a todas las observaciones que han sido realizadas a un prestamo de activo que un usuario
//en especifico han realizado
//recibe como parametros la conexion a la base de datos y el carnet de dicho usuario
public static function obtener_observaciones_activo($conexion, $codigo) {...28 lines }

//esta funcion es ocupada en la vista expediente_usuario que esta en la carpeta consulta, que se encuentra en la
//carpeta usuario
//es utilizada para listar a todas las observaciones que han sido realizadas a un prestamo de libros que un usuario
//en especifico han realizado
//recibe como parametros la conexion a la base de datos y el carnet de dicho usuario
public static function obtener_observaciones_libro($conexion, $codigo) {...27 lines }

//metodo utilizado en la vista alumnos_institucion, es utilizada para obtener el nombre de las instituciones
//y mostrarlas como leyenda de la grafica
//recibe como parametros la conexion a la base de datos y el codigo de la institucion
public static function nombre_institucion($conexion, $codigo) {...25 lines }

// esta funcion es utilizada en la vista restaurar_usuario, es utilizada para cambiar el estado de un
//usuario en especifico de inactivo a activo (estado = 1)
//recibe como parametros la conexion a la base de datos, el carnet del usuario a restaurar, y el nombre del usuario
public static function restaurar_usuario($conexion, $carnet, $nombre) {...58 lines }
```

2.2.17 Libros

```

class Repositorio_libros {
    //registra libros en la base de datos
    public static function insertarLibros($conexion, $libro, $a, $autores) {...50 lines }
    //retorna una lista de los libros disponibles
    public static function ListaLibros($conexion) {...33 lines }
    //retorna el catalogo de libros
    public function CatalogoLibros($conexion) {...33 lines }
    //lista de libros mas simple
    public function ListaLibros2($conexion) {...23 lines }
    //busca un libro por el codigo
    public static function BuscarLibro($conexion, $codigo) {...23 lines }
    //retorna lista de usuarios
    public static function BuscarUsuarios($conexion) {...12 lines }
    //retorna la informacion de un usuario segun su codigo
    public static function BuscarUsuario($conexion, $codigo) {...22 lines }
    //modifica la informacion de un libro
    public static function EditarLibro($conexion, $libro) {...32 lines }
    //cambia el estado de los libros a 1 que es dado de baja
    public static function DarBaja($conexion, $codigo, $motivo) {...27 lines }
    //obtiene la cantidad de libros de un mismo titulo
    public static function getCantidad($conexion, $codigo) {...19 lines }
    //obtiene la lista de libros del mismo titulo para dar baja
    public static function ListaDarBaja($conexion, $codigo) {...21 lines }
    //obtiene una lista de libros dados de baja
    public static function LibrosDadosBaja($conexion) {...29 lines }

    public static function LibrosDadosBaja2($conexion, $titulo) {...26 lines }

    //obitne la lista de libros danados
    public static function LibrosDanados($conexion) {...30 lines }

    public static function LibrosDanados2($conexion, $titulo) {...26 lines }
    //obtiene la lista de libros extraviados
    public static function LibrosExtraviados($conexion) {...29 lines }

    public static function LibrosExtraviados2($conexion, $titulo) {...26 lines }
    //obtiene la lista de los libros mas prestados
    public static function LibrosMasPrestados($conexion) {...27 lines }

    public static function LibrosMasPrestados2($conexion, $titulo) {...27 lines }
    //obitene la lista de los libros para imprimir el codigo de barras
    public static function CodigoBarras($conexion) {...13 lines }

    public static function CodigoBarras2($conexion, $titulo) {...13 lines }

    public static function insertar_bitacora($conexion, $accion) {...22 lines }

```

3 GLOSARIO

Término	Descripción
Bootstrap	Framework css que ayuda al diseño
Repositorios	Archivos que interactúan directamente con la base de datos para hacer las funciones de registrar, modificar y listar los datos